# Detecting People in Multi-view Environment using Height Map Reconstruction in Real-Time

Ákos Kiss

*Distributed Events Analysis Laboratory, Computer and Automation Research Institute (MTA SZTAKI), H-1111, Kende u. 13-17, Budapest, Hungary*

*Dept. of Control Engineering and Information Technology, Budapest University of Technology and Economics, H-1117 Budapest, Magyar tudósok körútja 2., Budapest, Hungary*

Tamás Szirányi

*Distributed Events Analysis Laboratory, MTA SZTAKI*

**Abstract**

In this article we address the problem of multiview detection: we detect pedestrians by their feet in scene space. Detection is based on searching cone intersections. The altitude of detection is also retrieved, which eliminates the need of planar ground - which is a common restriction in the related literature. We found positions can be computed accurately, and despite a large number of false positives, height map of the scene can be reconstructed with small error. Precision of the detector can be increased knowing height map, so that results of our method are comparable to SOA methods in case of planar ground - using somewhat different test methodology -, but adding the ability to handle arbitrary ground. Our algorithm is capable of real-time operation, based on two optimizations: decreasing number of cones, and approximating intersection bodies. Cones are back-projections of ellipses in images covering feet, this is ensured by filtering and clustering foreground pixels. Moreover, most demanding steps are parallelizable, and distributable due to lack of data dependencies.

*Keywords:* multi-view detection, 3D position, projection, real-time processing

## 1. INTRODUCTION

Single camera detecting and tracking relies on some kind of descriptors, like color, shape, and texture - a survey about state of the art in monocular pedestrian detection can be found in [3]. However, extraordinary and occluding

*Email addresses:* kiss.akos@sztaki.mta.hu (Ákos Kiss), sziranyi.tamas@sztaki.mta.hu (Tamás Szirányi)

objects are hard to detect. Multiple cameras are often used to overcome these limitations.

In these cases, consistency of object pixels among views will signal objects in 3D space. There are many methods for finding object pixels. Using stereo cameras, the disparity map can highlight foreground, or using wide-baseline stereo imaging, image-wise foreground detection is carried out.

The resulting set of object pixels can be further segmented, and consistency check might be reduced to likely correspondent segments. This might be done using color descriptors [14, 15], however color calibration [10] is necessary due to different sensor properties or illumination conditions.

Foreground masks can be projected to a ground plane, and overlapping pixels mark consistent regions [9]. Robustness can be improved using multiple parallel planes [12]. Furthermore, looking for certain patterns in the projection plane increases reliability [16].

In many works authors assume known homography between views and ground plane to carry out projection. In [7] homography parameters are estimated from co-motion statistics from multimodal input videos, eliminating the need of human supervision.

Projection of whole foreground masks is computationally expensive, but filtering pixels can reduce complexity. In some works, points associated with feet are searched, reducing foreground masks from arbitrary blobs to points and lines [13, 9].

Another gain of filtering foreground points is that - depending on the geometry - feet are less occluded than whole bodies, eliminating a great source of errors. Reducing occlusion is especially important in dense crowds, for example using top-view cameras [5].

## 2. OVERVIEW

Our goal was to design an algorithm for detecting people in a multiview environment with possibly many views where the geometry of the ground is arbitrary, which problem is not addressed in the related literature. Moreover we aimed at reaching real-time operation to make it available for surveillance systems.

We utilized the fact that foreground pixels correspond to lines in 3D scene space, and these lines intersect at surface points of the object. Finding such intersections however would require too many computations, as number of

line pairs $n_p$ is proportional to

$$n_p = (s_x s_y r_{fg})^2 \binom{n}{2} \tag{1}$$

where $s_x, s_y$ is the resolution of the camera image, $r_{fg}$ is the ratio of foreground pixels in image and $n$ is the number of views. We found that increasing number of views is not the bottleneck, but the great number of foreground pixels. Number of these pixels increases with resolution, and errors in foreground mask. Therefore we chose to

1. filter the foreground mask to find pixels relevant to detecting position of people, and
2. collect spatially coherent pixels and form one primitive from them instead of handling many lines.

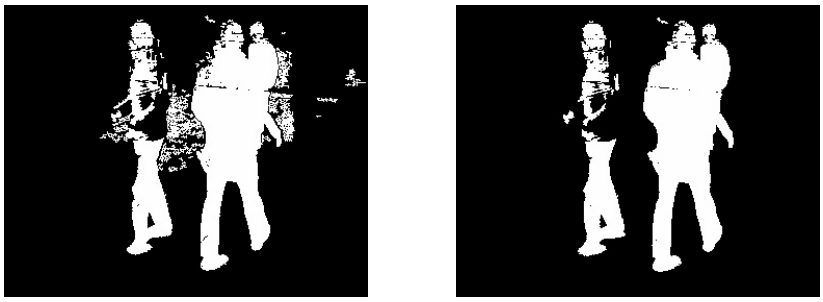
We filtered pixels possibly corresponding to feet, which we call candidate pixels. We covered these pixels with ellipses, these can be back-projected to cones in scene space. Intersecting a pair of cones replaces intersecting many line pairs.

Our approach has several advantages in means of both precision and speed:

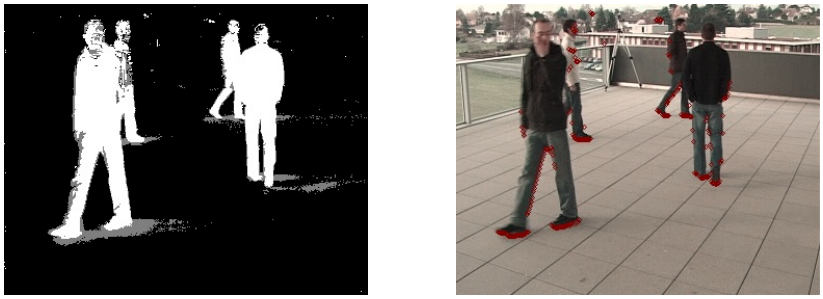
- for determining cone parameters, undistortion may be carried out with few computations leading to accurate parameters,
- number of cones is proportional to number of objects regardless of image resolution, this makes our cone matching approach scalable compared to pixel matching,
- ground doesn't have to be flat (unlike using homographies in [16, 12, 11, 9, 2]),
- we can compute synergy map without height range presumption like in [16, 12, 11].

On the downside:

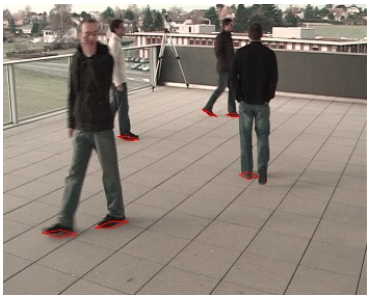
- our algorithm may fail on incorrect foreground mask. As we will show in Section 3.2, if foreground mask is incorrect, the extracted ellipses won't cover feet precisely, or won't even intersect any foot.



(a) Extracting foreground mask without (left) and with (right) white balance compensation.



(b) Extracting candidate pixels of feet from foreground mask.

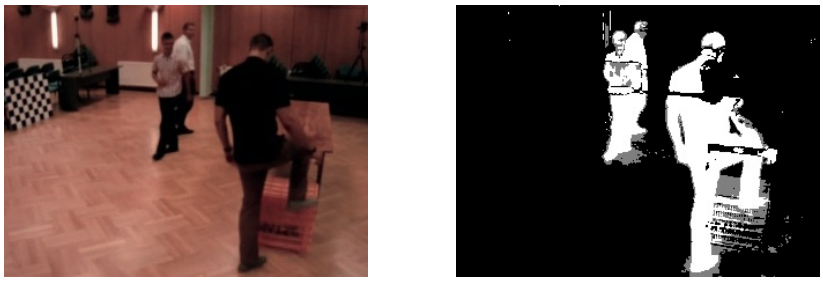


(c) Candidate pixels are clustered, clusters are modeled with ellipses (ellipses are visualized with lozenges).

Figure 1: Steps of extracting feet

- precise calibration is required for reliable estimation of cone parameters (both intrinsic and extrinsic parameters of the camera). A small error in cone or camera parameters can slightly change the direction of the cone, which results in a great error far from the camera.

We show applied preprocessing steps to form ellipses from foreground mask in section 3. Section 4 introduces steps of forming cones in scene space and matching these cones. Section 5 describes feet detection from cone matches



(a) Shadows may corrupt foreground detection even in carefully chosen color space. Foreground mask (right) for an input frame (left) is shown. A number of false positives are introduced due to corrupted foreground mask.

(b) Candidate pixels can appear on arms or on foreground artifacts, and also cones corresponding to different feet can intersect.

(c) Corresponding images showing time skew.

Figure 2: Sources of errors

as well as details of height map reconstruction. We show results and comparison to state of the art methods in Section 6 and conclude our work in Section 7.

## 3. PREPROCESSING

We found that camera's auto white balance function leads to incorrect foreground mask. This function aims at adopting to changes in lightning, but it is also affected by large objects entering or exiting the image, thereby changing white balance even if lighting conditions remain the same.

To neglect these effects, we extended the background model described in [1] with white balance adaptation. By computing and neglecting consistent changes in background pixel values, we managed to compensate changes in white balance. This resulted in a more reliable foreground detection (see Fig 1a)

Foreground detection was corrupted by shadows and reflections on the ground. We tried different color spaces to overcome this issue, and chose to use XYZ color space. It lead to best overall performance of our algorithm, however in certain situations foreground mask was still corrupted by shadows, as we can see in Fig 2a.

We filtered out small areas from the foreground mask to suppress noise. In our tests, we eliminated areas covering less than 7 pixels, as we found foreground blobs due to noise did not exceed this size. For other videos, this threshold might be adjusted according to video quality. Note that this step is not necessary, it only speeds up further processing.

### *3.1. Filtering foreground pixels*

We assumed that cameras are in upright position so that pixels of feet are bottom pixels of vertical lines in the foreground mask. We consider these pixels as candidates for feet, we call them candidate pixels. A sample of extracted candidate pixels can be seen in Fig 1b.

Note that using calibration information, images are easily rotated or search in a different upright direction may be carried out.

We clustered the candidate pixel set by connecting pixels closer than 3px. This is necessary, as candidate pixels are often not connected because of noise and steep edges. Note that these effects don't scale with resolution, so this threshold can be used for different image sizes, however, adjustment might be required for different image quality.

Clustering results in a set of pixel sets. We model each pixel set with an ellipse according to the spatial mean and variance of the pixels. This makes our algorithm robust to image resolution, as increasing resolution results in more candidate pixels, but the same number of clusters - thus reducing the number of 3D primitives, which drastically speeds up pairwise matching. Only the steps of preprocessing is affected by the resolution.

### *3.2. Forming Ellipses*

A cluster is considered to be a sampling of an elliptical area. For an ellipse with major and minor radius $a$ and $b$ parallel to axis, we know:

$$\int x^2 = \frac{a^3 b\pi}{4} \tag{2}$$

$$\int y^2 = \frac{ab^3\pi}{4} \tag{3}$$

$$\int xy = 0 \tag{4}$$

Rotating with $\alpha$ we get:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{5}$$

$$\int x'^2 = c^2 \int x^2 + s^2 \int y^2 \tag{6}$$

$$\int y'^2 = s^2 \int x^2 + c^2 \int y^2 \tag{7}$$

$$\int x'y' = sc \left( \int x^2 - \int y^2 \right) \tag{8}$$

$$tg2\alpha = \frac{2 \int x'y'}{\int y'^2 - \int x'^2} \tag{9}$$

$$\int x'^2 + \int y'^2 = \int x^2 + \int y^2 = \frac{ab\pi}{4}(a^2 + b^2) \tag{10}$$

$$\frac{\int x'y'}{sc} = \int x^2 - \int y^2 = \frac{ab\pi}{4}(a^2 - b^2) \tag{11}$$

$$\int 1 = ab\pi \tag{12}$$

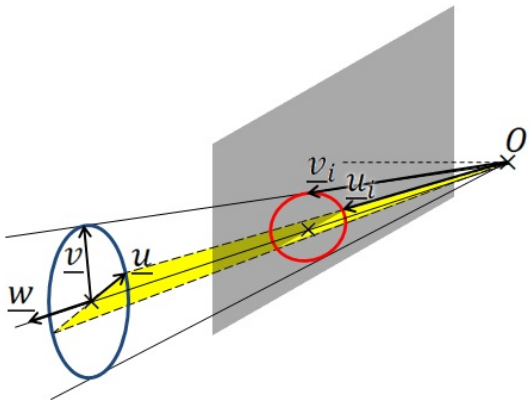
Where $s = sin\alpha$, $c = cos\alpha$. Now we can compute major and minor radius and rotation. We reject too small and upright ellipses. Minimal size can be adjusted for different setups and image resolution, in our experiments ellipses covering less than 7 candidate pixels were rejected, this ensured feet far from the camera are still detected. We defined maximal angle as 45°, foot direction does not depend on setup or resolution, so this threshold should work in any setups. Sample results can be seen in Fig. 1c.

## 4. DETECTING CORRESPONDENCES

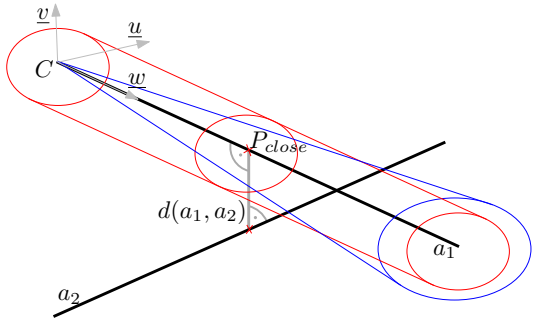
Cones corresponding to a foot will intersect close to the location of foot. However, cones corresponding to different feet may accidentally intersect leading to false detections. Moreover, candidate pixels can appear on arms or on foreground artifacts, introducing false cones. Fig 2b shows examples of such false matches.

### *4.1. Forming Cones*

We chose to describe a cone with a vertex $C$ and three orthogonal vectors $\underline{u}$, $\underline{v}$ and $\underline{w}$, where $\underline{w}$ is the unit length




(a) Model of a cone. Vertex $C$ is in the projection center $O$, $\underline{w}$ points toward center of ellipse in image plane. Bevel angles and axes directions ($\underline{u}$, $\underline{v}$) are computed from axes in image plane ($\underline{u}_i$, $\underline{v}_i$).




(b) Cylinder for cone matching.

Figure 3: Steps of extracting feet

direction vector of axis and $\underline{u}$, $\underline{v}$ are direction vectors of major and minor axes (Fig. 3a). Bevel angles are determined by the length of $\underline{u}$, $\underline{v}$ vectors. Cone consists of points $\underline{p}$ where

$$\begin{aligned} \big((\underline{p}-C)\underline{u}\big)^2 + \big((\underline{p}-C)\underline{v}\big)^2 \leq \big((\underline{p}-C)\underline{w}\big)^2, \\ (\underline{p}-C)\underline{w} \geq 0 \end{aligned} \tag{13}$$

From extrinsic parameters of camera we know 3D position of points of image plane as well as the optical center $O$. $\underline{w}$ points from $O$ to the center of the ellipse, and $\overrightarrow{Ou_i}$, $\overrightarrow{Ov_i}$ are in symmetry planes, where $u_i$ and $v_i$ are extremal points of ellipse in image plane (Fig 3a). We determined major and minor axes directions by vector products to ensure orthogonality:

$$\underline{v} = \underline{u_i} \times \underline{w} \tag{14}$$

$$\underline{u} = \underline{w} \times \underline{v} \tag{15}$$

Finally, $\underline{u}$ and $\underline{v}$ are scaled according to major and minor bevel angles so that (13) stands.

### *4.2. Cone Matching*

Intersection of cones is a complex body, especially when bevel-angles differ for major and minor axes - apart from some extraordinary situations. Determining this body would be difficult, but our experiments showed that we don't need the exact intersection, an approximate solution is sufficient. We applied simple heuristics for finding cone intersections - which we call matches. We simplified computation in two ways:

1. generally a foot covers a small area of the image, so cones will have small bevel-angles. Consequently in a relatively small space segment, they can be replaced by elliptical cylinders. We assume that intersection will take place near the point where the two cone axes are closest, so we compute major and minor radii at this offset ($\overrightarrow{CP_{close}}$ in Fig. 3b).
2. Computing exact intersection of cylinders is still a hard problem, and the intersection body is still complex. Therefore instead of computing intersection body, we tried to find an optimal point $\underline{p}$ in space, for which distance from axes of the elliptical cylinder is minimal - considering different radii.

After computing $|\overrightarrow{CP_{close}}|$ distance, we scale $\underline{u}$ and $\underline{v}$ vectors to match major and minor cylinder radii:

$$\begin{aligned} \underline{u'} &= \underline{u}|\overrightarrow{CP_{close}}| \\ \underline{v'} &= \underline{v}|\overrightarrow{CP_{close}}| \end{aligned} \tag{16}$$

Now, equation of the elliptical cylinder becomes

$$\left((\underline{p}-C)\underline{u'}\right)^2 + \left((\underline{p}-C)\underline{v'}\right)^2 \leq 1 \tag{17}$$

Distance from axis - left side of (17) - is a linear function of $\underline{p}$, enabling us to write a linear equation system expressing $\underline{p}$ is on both axes (index refers to cylinders):

$$\begin{aligned}\left((\underline{p}-C_1)\underline{u}_1'\right)^2+\left((\underline{p}-C_1)\underline{v}_1'\right)^2=0\\ \left((\underline{p}-C_2)\underline{u}_2'\right)^2+\left((\underline{p}-C_2)\underline{v}_2'\right)^2=0\end{aligned} \tag{18}$$

equivalent to

$$\begin{bmatrix}\underline{u}_1'^T\\ \underline{v}_1'^T\\ \underline{u}_2'^T\\ \underline{v}_2'^T\end{bmatrix}\underline{p}=\begin{pmatrix}\underline{u}_1'C_1\\ \underline{v}_1'C_1\\ \underline{u}_2'C_2\\ \underline{v}_2'C_2\end{pmatrix} \tag{19}$$

$$A\underline{p}=\underline{b} \tag{20}$$

Of course, axes will practically never intersect, only approximate solution is possible. Solving subject to least square error is straightforward and practical in our case. It is straightforward, because it is a distance from axes considering different major and minor radii, and it is practical, because this optimization is easy to carry out - for example using pseudo inverse.

If $\underline{p}$ is outside any cylinder, we conclude cones are not intersecting, otherwise, they intersect and $\underline{p}$ is the position of the intersection - which we call match. We also assign a weight to matches computed from the error of the approximate solution.

## 5. DETECTING FEET

### *5.1. Detection*

One match itself can be a false detection. However, for two matches, foot has to be visible in at least three views, which is not common in dense crowd. Therefore we applied a threshold to the weight of matches to decide if foot is detected.

We merged matches close in space to avoid multiple detections. We also assigned weight to such sets, computed from the weights of the matches in a way, that the possibility of detection highly increases with the number of matches.

To ease merging close matches, we merged matches over a grid on horizontal plane. As feet don't appear above each other, in a cell of the grid we don't mix matches at different altitudes. We chose to collect matches in blocks, which consist of overlapping $2 \times 2$ cells, then compute the weight of match sets in every block. Using overlapping blocks is necessary to achieve smooth weight map over the grid.

Every local maximum in this 2D weight map with a weight above a given threshold is considered a detected foot. This threshold is a parameter of our algorithm, higher threshold leads to increased precision along with decreased recall. 3D position of foot is computed as the barycenter of matches in the block, where $z$ coordinate is height (parallel to upright direction).

### *5.2. Height Map*

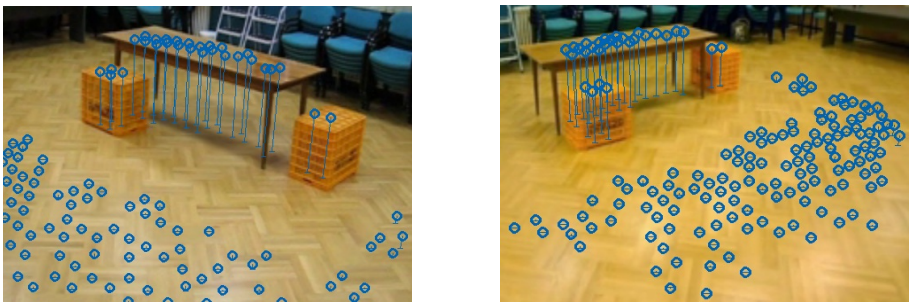
As we mentioned before, and showed in Fig 2b, false detection may occur due to incorrect foreground detection or random intersections of non-corresponding cones. We found that in a dense crowd many false detections appear. However the height of these detections is quite random, and they usually occur far from the ground.

Consequently these false positives appear as a noise. To suppress this noise, we aggregated all detections from a test set and applied statistical filtering with normal distribution presumption. As a result, we got positions on ground with high confidence. These dense points will be height points of ground.

The phrase height map is somewhat misleading, because at the border of surfaces at different altitude, it is possible to have height points above each other, however in our tests this never occurred. Thereby we kept the height map expression.

After extracting height map, we can ignore detections far from any height point, leading to a much more reliable method. We speeded up height point searching by accumulating detections in a 3D grid, which makes periodic rerun possible. This enables us to refine height map as more and more detections occur.

A sample height map of our non planar test case is shown in Fig 4a. There are certain areas where few detections took place, this led to incomplete height map.



(a) Generated height map points for our test case.

(b) Corresponding frames from different cameras with the detected feet marked

Figure 4: Sample results of detection and determining height map.

## 6. EXPERIMENTS

We tested our algorithm on a commonly used test sequence for multiview detection, the *EPFL terrace* sequence [4] (sample results can be seen in Fig 4b), and our own test videos (*SZTAKI* sequence).

We made test videos of a scene where three planar surfaces are present in different heights. This was required to demonstrate capabilities of our method as available multiview test sets present planar ground. We recorded videos

from 4 views with different kind of consumer digital cameras with video capture function. We found that our algorithm performed well despite the different camera parameters, distortion and image quality.

Our approach, detecting foot has advantages and disadvantages as well. Feet are always near ground, which makes height map detection possible and helps rejecting false positives. Also, for certain camera setups, feet are less likely to be occluded compared to whole bodies.

On the other hand in case of detecting foot positions, precise time synchronization is necessary, as a small time skew can corrupt detection. In Fig. 2c, we can see the moving leg is bended in one view while straight in the other. Hardly noticeable in the small image, however the distance in space is greater than the size of the foot itself.

### *6.1. Height Map Reconstruction*

We reconstructed height map for both *EPFL* and *SZTAKI* sequences. We found that height map reconstruction was carried out with high precision in both cases.

#### *6.1.1. EPFL sequence*

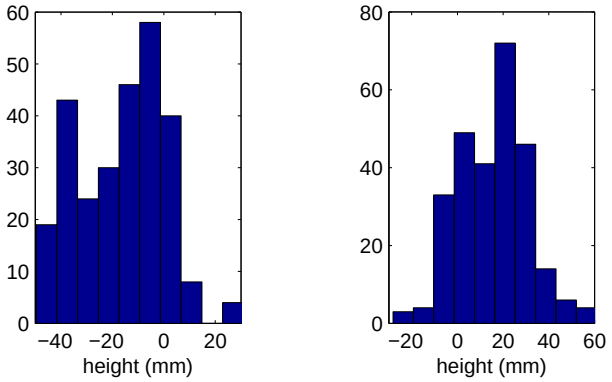
In case of *EPFL* sequence, we measured that the reconstructed ground is a flat plane with small error: $\mu = -1.6cm$ (mean), $\sigma = 1.7cm$ (std. deviation). Fig 5a (left histogram) shows the histogram of height values.

We measured distribution different from normal distribution - which we expected assuming noise-like error. A possible source of systematic error is incorrect plane normal, so we computed optimal normal vector by fitting a plane on height points. The optimal normal vector was very close to upright direction specified by camera calibration. It diverged by only $0.36^\circ$, resulting in slightly smaller error: $\mu = 1.6cm$, $\sigma = 1.5cm$. However, using this normal led to a significantly better distribution (Fig 5a, right histogram).

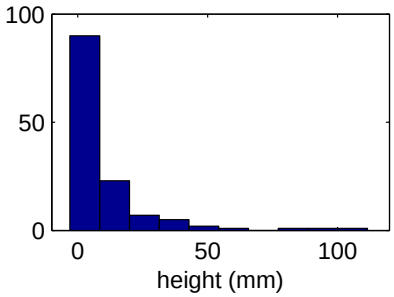
#### *6.1.2. SZTAKI sequence*

In case of our sequence, we found that all three surfaces were found, and their height was determined with high accuracy. Measurements are summarized in Table 1a. Fig 5b shows the histogram of heights for points on floor.

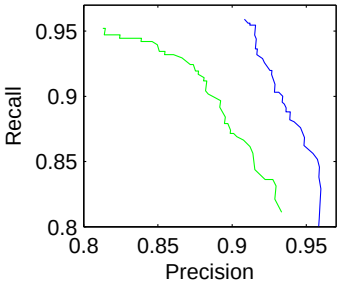
There are 3 outlier points (above 6cm high) for the floor. These appear on the edge of the area of interest, as can be seen in Fig 4a. We found this is due to sagging people - which lead to false candidate pixels on image borders. Without these outlier points, mean and $\sigma$ for floor becomes 0.9cm and 1.1cm respectively.




(a) Histogram of height of ground points for *EPFL* dataset in reference coordinate system - according to calibration data (left), and in direction of computed normal vector (right).




(b) Histogram of height of floor points for *SZTAKI* dataset.




(c) ROC curve measured in function of detection threshold on *EPFL* (blue), and *SZTAKI* (green) sequences.

Figure 5: Evaluation of detection and height point calculation.

### *6.2. Detecting People*

We tested our method using manually created ground truth information of feet positions. We implemented an application to create ground truth, in which we have to mark a foot in at least 2 views and its position is determined by solving a linear equation similar to (19). If we mark foot in more views, number of rows of $A$ and length of $\underline{b}$ increases.

We collected feet positions frame by frame, person by person. 1 or 2 legs can be specified for a person, because sometimes only one foot is visible from more views.

For the evaluation, we defined a region of interest (ROI), a rectangle on the ground plane and matched detections to ground truth inside this area. In case of *EPFL* dataset, this rectangle is specified along with the dataset, for our test case this area was chosen so that every part is visible from at least 3 views.

Table 1: Numerical evaluation of two main aspects: accuracy and running time.

(a) Statistical information on surfaces found in scene.

| surface | height | $\mu$ | $\sigma$ | # of points | min | max |
|---|---|---|---|---|---|---|
| floor | 0cm | 1.1cm | 1.7cm | 131 | -0.3cm | 11.2cm |
| box | 50cm | 50.2cm | 0.6cm | 6 | 49.1cm | 50.8cm |
| table top | 73cm | 74.4cm | 1.2cm | 23 | 70.3cm | 75.9cm |

(b) Average processing time of each step for all 4 views (single threaded implementation, 2.4GHz Core 2 Quad CPU). Foreground detection, filtering and cone forming can be done parallel (per camera), cone matching and detection requires all cone information to be present.

| test set | foreground detection | candidate filtering | cone forming | $\sum$ (per camera) | cone matching | detection | $\sum$ (per system) |
|---|---|---|---|---|---|---|---|
| *EPFL* | 51.2ms | 3.37ms | 63us | 13.7ms | 879us | 28us | 907us |
| *SZTAKI* | 32.5ms | 2.88ms | 1.99ms | 9.35ms | 600us | 18us | 618us |

We accepted a detection as true positive, if it was not further than 25cm from a ground truth foot position. This is approximately the length of a foot. Afterwards we eliminated people and detections outside the ROI, and computed false negatives and false positives from left people and detections accordingly.

As mentioned in Section 5.1, a detection threshold is applied to the weight map. Trade off between precision and recall is balanced with this threshold. To evaluate our method, we ran the test with different threshold values, resulting in precision-recall curves. Results are shown in Fig 5c.

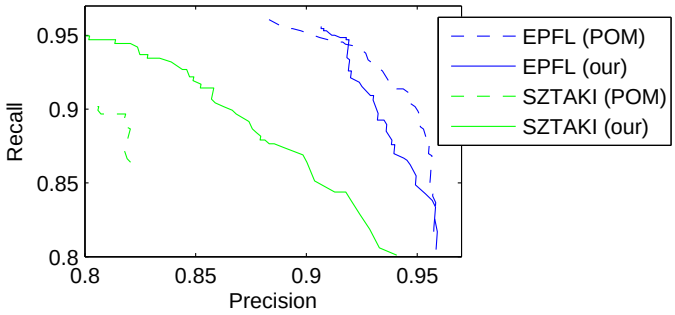
#### *6.2.1. Comparison to SoA Method*

We compared our method to two SoA methods referred to as POM[6] and 3DMPP[16]. Comparison is not straightforward, as these methods detect bodies projected to a ground plane, in contrast we detect feet positions in space. Consequently, a different evaluation criteria had to be used: we consider someone detected if either of his legs were detected.

Fig 6a shows results found in [16] for POM and 3DMPP - evaluated on EPFL and PETS datasets - compared to

| | Precision | Recall | Time (without foreground detection) |
|---|---|---|---|
| POM | 87.20% | 95.56% | ∼0.5s |
| 3DMPP | 97.5% | 95.5% | ∼1s |
| our | 90.66% | 95.61% | ∼0.01s |

(a) Precision/recall values measured on EPFL dataset. Results for POM and 3DMPP are copied from [16].




(b) ROC curve measured in function of detection threshold on *EPFL* (blue), and *SZTAKI* (green) sequences.

Figure 6: Comparing our results to a SoA methods.

our measurements - evaluated only on EPFL dataset. We used similar amount of test data (179 frames - every 25th -, 661 objects), and the same ROI. Results show that our method is comparable to these SoA methods in case of flat ground.

We also compared results on our dataset to show effect of non-planar ground. For this we chose POM as our framework was easy to generate input for it's available implementation.

For our dataset, foreground detection was challenging due to reflective ground and poor video quality. As a result, both methods performed worse in this case, however our algorithm performed noticeably better (see Fig 6b).

### *6.3. Running Time*

We measured average running time of processing steps, results are shown in Table 1b. Every step highly depends on input data, thus ratio of foreground mask processing times differ from ratio of pixels. In these cases, the video resolution was $360 \times 288$ for *EPFL* and $320 \times 240$ for *SZTAKI* sequences

Results show that our method is capable of real-time processing of 4 views with $360 \times 288$ resolution even with single threaded implementation. However, parallel processing is also possible with some restrictions.

Several steps of processing perform on single images, therefore can be parallelized. Clearly, foreground detection, filtering, ellipse and cone forming is done imagewise. These steps can be done even on different computers, or on smart cameras, as no data exchange is required between views at all.

Cone matching and detection requires all cone information to be present. Due to short processing time, these steps should run sequentially on a dedicated resource. In a camera network, where not all cameras have overlapping field

of view, camera sets can be processed individually.

In case of above mentioned SOA methods, mean processing time of foreground mask is in the order of $0.5s$ for POM, and $1s$ for 3DMPP, using unoptimized code on similar processor (data provided by author of [16]). However in our case, processing foreground took less than $10ms$.

Moreover, many works - including 3DMPP - operate projecting parts or whole foreground masks to planes, which makes distributed computing impossible due to data dependencies.

## 7. CONCLUSION

We proposed a multiview-detection algorithm that retracts 3D position of people using multiple calibrated and synchronized views. In our case, unlike other algorithms, non-planar ground can be present. This is done by modeling possible positions of feet with 3D primitives, cones in scene space and searching for intersections of these cones.

For good precision, height map of ground should be known. Our method can compute height map on the fly, reaching high precision after a startup time.

After height map detection we measured precision and recall values comparable to SOA methods on a commonly used data set. Our algorithm worked well also on our test videos we made to demonstrate capabilities of handling non-planar ground.

In the future we plan to examine tracking people by their leaning leg positions[8].

## 8. ACKNOWLEDGE

The authors would like to thank François Fleuret for making implementation of POM available, and Ákos Utasi for sharing his experiments of evaluating POM and 3DMPP algorithms - his code is also available for download.

This work has been supported by the Hungarian Scientific Research Fund grant OTKA #106374.

## 9. REFERENCES

[1] Benedek, C. and Szirányi, T. (2008). Bayesian foreground and shadow detection in uncertain frame rate surveillance videos. *IEEE Image Processing*, 17(4):608–621.

[2] Berclaz, J., Fleuret, F., and Fua, P. (2006). Robust people tracking with global trajectory optimization. In *IEEE CVPR*, pages 744–750.

[3] Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):743–761.

[4] EPFL (2011). Multi-camera pedestrian videos. http://cvlab.epfl.ch/data/pom/.

[5] Eshel, R. and Moses, Y. (2010). Tracking in a dense crowd using multiple cameras. *International Journal of Computer Vision*, 88:129–143.

[6] Fleuret, F., Berclaz, J., Lengagne, R., and Fua, P. (2008). Multicamera people tracking with a probabilistic occupancy map. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):267–282.

[7] Havasi, L. and Szlavik, Z. (2011). A method for object localization in a multiview multimodal camera system. In *CVPRW*, pages 96–103.

[8] Havasi, L., Szlávik, Z., and Szirányi, T. (2007). Detection of gait characteristics for scene registration in video surveillance system. *IEEE Image Processing*, 16(2):503–510.

[9] Iwase, S. and Saito, H. (2004). Parallel tracking of all soccer players by integrating detected positions in multiple view images. In *IEEE ICPR*, pages 751–754.

[10] Jeong, K. and Jaynes, C. (2008). Object matching in disjoint cameras using a color transfer approach. *Machine Vision and Applications*, 19:443–455.

[11] Khan, S. and Shah, M. (2006). A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *ECCV 2006*, Lecture Notes in Computer Science, pages 133–146.

[12] Khan, S. and Shah, M. (2009). Tracking multiple occluding people by localizing on multiple scene planes. *PAMI*, 31(3):505 –519.

[13] Kim, K. and Davis, L. S. (2006). Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In *ECCV*, pages 98–109.

[14] Mittal, A. and Davis, L. (2001). Unified multi-camera detection and tracking using region-matching. In *IEEE Multi-Object Tracking*, pages 3 –10.

[15] Mittal, A. and Davis, L. (2002). M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *ECCV 2002*, pages 18–33.

[16] Utasi, Á. and Benedek, C. (2011). A 3-D marked point process model for multi-view people detection. In *IEEE CVPR*, pages 3385–3392.